GP51A-1072

SpF: Enabling Petascale Performance for Pseudospectral Dynamo ls

T. Clune[1], W. Jiang[2,1], J. Vriesema[3] and G. Gutmann[4]

[1] NASA Goddard Space Flight Center, [2] SGT, [3] University of Arizona, [4] Tokyo Institute of Technology

Introduction

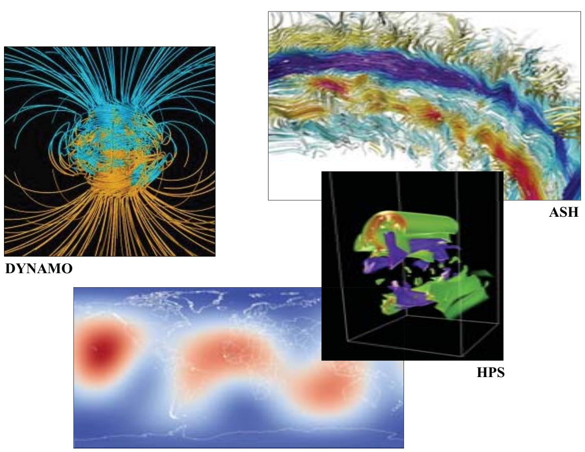
Pseudospectral (PS) methods possess a number of characteristics (e.g., efficiency, accuracy, natural boundary conditions) that are extremely desirable for dynamo models. Unfortunately, dynamo models based upon PS methods face a number of daunting challenges, which include exposing additional parallelism, leveraging hardware accelerators, exploiting hybrid parallelism, and improving the scalability of global memory transposes. Although these issues are a concern for most models, solutions for PS methods tend to require far more pervasive changes to underlying data and control structures. Furthermore, performance improvements in one model are difficult to transfer to other models, resulting in significant duplication of effort across the research community.

We have developed SpF, an extensible software framework for PS methods, which is intended to enable extreme scalability and optimal performance. High-level abstractions provided by SpF unburden applications of the responsibility of managing domain decomposition and load balance while reducing the code changes required to adapt to new computing architectures. The key design concept in SpF is that each phase of the numerical calculation is partitioned into disjoint numerical "kernels" that can be performed entirely in-processor. The granularity of domain-decomposition provided by SpF is only constrained by the data-locality requirements of these kernels. SpF builds on top of optimized vendor libraries for common numerical operations such as transforms and matrix solvers, but can also be configured to use open source alternatives for portability. SpF includes several alternative schemes for global data redistribution and is expected to serve as a testbed for further research into optimal approaches for advanced network architectures.

Goals

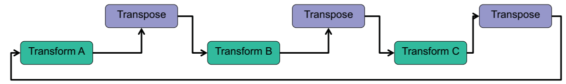
- Scalable and Efficient
  - Extremely fine-grained parallelism
  - Multilevel parallelism (MPI+OMP)
  - Isolate comp. intensive sections
  - Utilize optimized vendor libraries
  - Parallel I/O
- Flexible
  - Variant geometry
  - Alternative radial schemes
  - Variant load balance strategies
- Reduced duplication
  - Shared code infrastructure
  - Exchange of innovations
- Extensible – OO design
- Robust and Portable
  - Test driven development
  - Nightly automated regression tests

Targeted NASA Applications



Design Elements

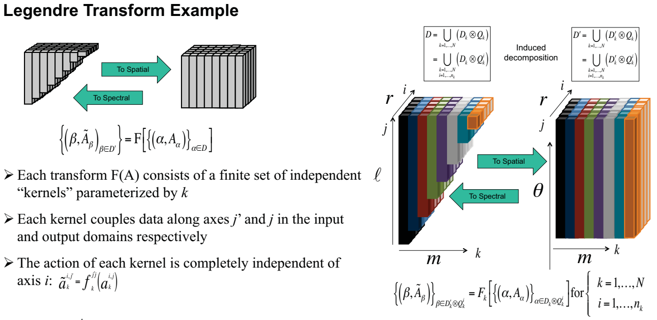
Parallel implementations of PS methods can generally be implemented as a loop consisting of alternating series of numerical transforms and global memory transposes/permutations.



Typical applications use inflexible, parameterized heuristics for decomposing the data across processes for each transform. Transposes are then hardcoded to align with these built-in assumptions. This approach leads to brittle implementations that require significant revision when undergoing further optimization.

The key innovation in SpF is to enable implementation of transforms in a *decomposition-independent, communication-free* manner. As demonstrated below for Legendre transforms, locality properties of transforms lead to a natural 3D parameterization of the input and output domains. The numerical operation only couples values along one of these dimensions, which allows a completely arbitrary distribution of the remaining axes.

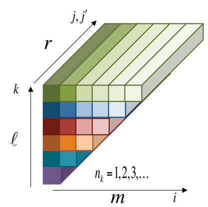
Legendre Transform Example



$$\left\{(\beta,\tilde{A}_\beta)_{\beta\in D'}\right\} = F\left[\left\{(\alpha,A_\alpha)\right\}_{\alpha\in D}\right]$$

- Each transform F(A) consists of a finite set of independent "kernels" parameterized by *k*
- Each kernel couples data along axes *j'* and *j* in the input and output domains respectively
- The action of each kernel is completely independent of axis *i*: $\tilde{a}_i^{j'} = f_k^{jj'}(a_i^j)$

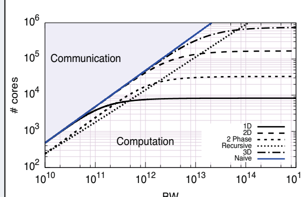
$$\left\{(\beta,\tilde{A}_\beta)\right\}_{\beta\in D_k'\otimes Q_i} = F_k\left[\left\{(\alpha,A_\alpha)\right\}_{\alpha\in D_k\otimes Q_i}\right] \text{for} \begin{cases} k=1,\dots,N \\ i=1,\dots,n_k \end{cases}$$



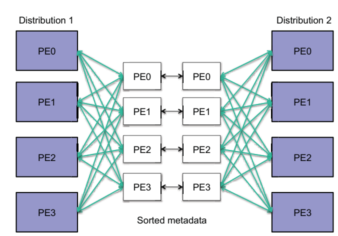
The software framework is free to assign each "pencil" (*i*,*k*) to an arbitrary processor. Secondary considerations such as memory footprint and cache-locality lead to prioritizing distribution along the kernel axis. Early parallel implementations of PS applications generally used 1D decompositions corresponding to just one of these axes. Modern implementations often exploit a 2D decomposition, but still cannot fully exploit parallelism when n_k is is not constant such as for the Poisson solver illustrated by the figure to the left.

Automated Permutations

To support the arbitrary domain decompositions, SpF must fully automate permutations between alternative layouts.



At extreme scale, the efficient implementation of data permutations is nontrivial. The figure above shows the benefit of multi-staged permutations for a typical petascale dynamo simulation. SpF enables exploring such alternatives with minimal effort.



Naïve initialization requires $O(N^2)$ operations, which is prohibitive for petascale applications. SpF uses a novel approach based upon parallel sort to achieve a complexity of O(N log N).

Software Framework Implementation

Key Abs s

- Algorit
  - Ker composable "atom" of a numerical phase
  - Ker generates parameterized set of kernels and associated in/out Field objects
- Integra ns integration in time (RK, AB, CN, etc.)
- Data
  - Ind gebra for composing descriptions of arbitrary domains
  - Fiel r for data associated with an IndexSpace
  - Dat s multiple parallel I/O formats (initially just pHDF5)
- Comm d Load Balancing
  - Per tributes data from one Field to another
  - Dis s metadata from KernelFactory to perform domain decomposition

Test drive nt (TDD) has been routinely applied using pFUnit 2.0 to ensure reliability and portab asic framework.

Index S resentation of Computational Domains

A

i
1
2
3

$A\otimes B$

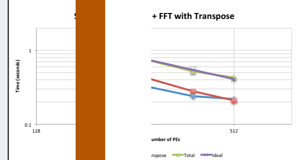
i	j
1	3
2	3
3	3
1	5
2	5
3	5

$B\oplus B'$

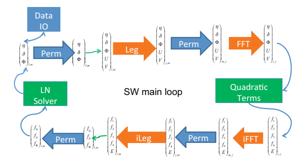
j
3
5
2
4

Large irregular computational domains are specified using a simple algebra of outer products and direct sums on 1D "axes." Applications exchange data with the framework using these structures to specify layout in memory.

Current



The above onstrates the scalability of a synth lication that combines a Legendre FFT, and a transpose.



In this schematic of an SpF-based SW, customization is only required for the items indicated in green.

Next Steps

The basic rk is now mostly complete, and work has begun on porting legacy application m, the primary focus will be to extend SpF in the following ways:

- Adv hms for Distributor and Permutor subclasses for extreme scales
- Facto ucting linear systems for alternative radial schemes
- Incor vare accelerators
- Distr ls – large kernels with internal parallelism (e.g., implicit coriolis)

References

Beck, K., Development: by Example, 2002.
Brummel, netration and Overshooting in Turbulent Compressible Convection", *ApJ.* 570:825-8
Clune, T., an Unit Testing Framework (pFUnit) http://sourceforge.net/p/pfunit/_list/git
Glatzmaier ne, T., "Computational Aspects of Geodynnamo Simulations", *Comput. Sci. Eng.* 2:61-
Hack, J. an "Description of a Global Shallow Water Model Based on the Spectral Transform CAR Technical Note, 1992.
Jiang, W. a ., "An MPI-based MoSST core dynamics model", *Physics of the Earth and Planetary*) (2008), 46–51.
Miesch, M e-dimensional spherical simulations of solar convection. I. Differential rotation an lution achieved with laminar and turbulent states", *ApJ*, 2000.